# Stream-based Computation in Monoidal Categories

Elena Di Lavore[1], Giovanni de Felice[2], and Mario Román[1]*

[1] Tallinn University of Technology
[2] University of Oxford

**Abstract**

We construct a category of stream transducers over an arbitrary monoidal category and we characterize it in terms of final coalgebras. Stream transducers capture the notion of an iterated process which holds memory states $M_t$ and reacts to streams of inputs $X_t$ by producing outputs $Y_t$. In the cartesian setting, they coincide with the well-studied notion of causal function on streams. In the probabilistic setting, they correspond to controlled stochastic processes.

## 1 Introduction

Monoidal categories provide an algebra of processes that compose sequentially and in parallel. This algebra is abstract enough that it can deal with multiple *paradigms of computation* in a unified fashion. The category of sets and functions gives semantics to the typed lambda calculus [16]. The category of sets and relations formalises database queries [3]. Categories of Markov kernels are used for probabilistic programming [6, 10], Hilbert spaces and linear maps for quantum computing [1]. Moreover, monoidal categories have a practical graphical calculus which allows us to reason about parallel programs formally using string diagrams [14, 24].

Stream-based (or dataflow) computation [12, 28] deals with processes that, at each time $t = 0, 1, \ldots$, receive an input $X_t$ and produce an output $Y_t$. The processes must be causal: the output $Y_t$ can only depend on the inputs $X_0, \ldots, X_t$. Dataflow programming has applications in control systems [12], data science [20] and text processing [9], among others.

The coalgebraic approach to stream-based computation [13, 27] defines stream functions as coKleisli morphisms of the list comonad $\mathbf{List}^+ \colon \mathbb{C}^{\mathbb{N}} \to \mathbb{C}^{\mathbb{N}}$ defined on objects by $\mathbf{List}^+(\mathbf{X})_n \coloneqq \prod_{t=0}^{n} X_t$. However, it is not trivial to extend this comonad to other important non-cartesian categories such as the category of stochastic functions: a counit and comultiplication for this comonad are natural if and only if the monoidal product is cartesian.

The question, thus, becomes: *given a paradigm of computation represented by a monoidal category, how to construct its corresponding stream-based paradigm?*

**Contributions.** Our first contribution is the definition of the category of stream transducers $\mathbf{Stream}_{\mathbb{C}}$ over any monoidal category $(\mathbb{C}, \otimes)$. These capture the notion of an iterated process which holds memory states $M_t$ and reacts to streams of inputs $X_t$ by producing outputs $Y_t$. Our second contribution is the characterisation of the hom-sets of $\mathbf{Stream}_{\mathbb{C}}$ as final coalgebras. In cartesian categories, our construction recovers the coalgebraic definitions of [25, 27]. In the category of stochastic functions, our construction captures the notion of discrete stochastic process [23].

*Elena Di Lavore, and Mario Román were supported by the European Union through the ESF Estonian IT Academy research measure (project 2014-2020.4.05.19-0001).

**Related work.** The cartesian case has been studied extensively using coalgebras [13, 27]. Our approach generalizes that of [25] and relates it to the coalgebraic approach [11,21,27]. The definition of stream transducers already appeared in [22], where they were called *infinite combs*.

## 2 Stream transducers

A **stream transducer** is a process described by a sequence of morphisms $f_0, f_1, f_2 \ldots$ that represent its action $f_t \colon M_{t-1} \otimes X_t \to Y_t \otimes M_t$ at time $t = 0, 1, 2, \ldots$ At each step $t \in \mathbb{N}$, the stream transducer takes an input $X_t$ and, together with the stored memory $M_{t-1}$, produces some output $Y_t$ and writes to the memory $M_t$. The memory is initially empty, with $M_{-1} \coloneqq I$ being the unit of the monoidal category.

**Definition 2.1** (Stream transducers)**.** Let $(\mathbb{C}, \otimes, I)$ be a monoidal category. A *stream transducer* between two sequences of objects in $\mathbb{C}$ representing inputs $X_0, X_1, \ldots$ and outputs $Y_0, Y_1, \ldots$, is a sequence of objects $M_0, M_1, \ldots$ together with a sequence of morphisms

$$\langle M_n \mid f_n \colon M_{n-1} \otimes X_n \to Y_n \otimes M_n \rangle_{n \in \mathbb{N}} \quad \text{where, by convention, } M_{-1} \coloneqq I.$$

We say two transducers are equal when, for every natural number $n \in \mathbb{N}$, they are equal under the equivalence relation generated by $\langle M_t \mid (h_{t-1} \otimes \mathrm{id}); f_t \rangle = \langle M'_t \mid f_t; (\mathrm{id} \otimes h_t) \rangle$ for any $h_t \colon M_t \to M'_t$ and $f_t \colon X_t \otimes M'_{t-1} \to Y_t \otimes M_t$, with $t = 0, \ldots, n$.

**Proposition 2.2** (see [22])**.** *Stream transducers over a monoidal category $(\mathbb{C}, \otimes, I)$ form a category $\mathbf{Stream}_{\mathbb{C}}$. This is, moreover, a symmetric monoidal category with feedback (as in [8, 15, 26]) when $\mathbb{C}$ is symmetric.*

The full construction is detailed in [22], where stream transducers are called *infinite combs*. The cartesian case was first studied by Sprunger and Katsumata [25], who called them *stateful morphism sequences*.

**Coalgebraic characterisation.** Classically, type-variant streams have a neat coinductive definition that says *"a stream with types $\mathbf{A} = A_0, A_1, A_2, \ldots$ is an element of $A_0$ together with a stream with types $A_1, A_2, A_3, \ldots$"*. That is, streams are the greatest fixpoint of the equation of functors $\mathbf{Str} = \mathbf{Id} \times \mathbf{Str}$, which expands to $\mathbf{Str}(A_0, A_1, \ldots) = A_0 \times \mathbf{Str}(A_1, A_2, \ldots)$. This fixpoint is then computed to be $\mathbf{Str}(\mathbf{A}) = \prod_{n \in \mathbb{N}}^{\infty} A_n$.

In the same vein, *"a stream transducer from $\mathbf{X} = X_0, X_1, \ldots$ to $\mathbf{Y} = Y_0, Y_1, \ldots$ is a process from $X_0$ to $Y_0$ communicating along a channel with a stream transducer from $X_1, X_2, \ldots$ to $Y_1, Y_2, \ldots$"*. That is, stream transducers are the greatest fixpoint of the equation of profunctors $\mathbf{Stream} = \mathbf{hom} \odot \mathbf{Stream}$, where the binary operation $\odot$ describes composition along a channel. The above equation expands to

$$\mathbf{Stream}(\mathbf{X}; \mathbf{Y}) = \int^{M} \mathbf{hom}(X_0, Y_0 \otimes M) \times \mathbf{Stream}(M \otimes X_1, X_2, \ldots; Y_1, Y_2, \ldots). \tag{1}$$

We can then compute an explicit formula for stream transducers applying Adamek's theorem [2].

**Theorem 2.3.** *The set of stream transducers from $X_0, X_1, \ldots$ to $Y_0, Y_1, \ldots$ is the greatest fixpoint of Equation (1). This fixpoint is explicitly given by*

$$\mathbf{Stream}(X_0, X_1, \ldots; Y_0, Y_1, \ldots) = \lim_{n} \int^{M_0, \ldots, M_n} \prod_{t=0}^{n} \mathbf{hom}(X_t \otimes M_{t-1}, Y_t \otimes M_t) \tag{2}$$

*where, by convention, $M_{-1} \coloneqq I$ is the monoidal unit.*

The formula in Equation (2) deserves an explanation. We are describing a process in $n$ stages, and letting $n$ go to infinity as we take a limit. The integral sign is a coend [18, 19], which can be read as an existential quantifier. A representative element of this coend is a list of $n$ morphisms $f_t : X_t \otimes M_{t-1} \to Y_t \otimes M_t$ for some choice of $n$ 'memory channels' $M_0, \dots, M_n$, which are objects of $\mathbb{C}$.

# 3 Examples

**Cartesian streams.** Our first example serves as a sanity check: in a cartesian monoidal category $\mathbb{C}$, our definition recovers the usual notion of stream [25, 27]. Indeed, in the cartesian case, the universal property of the cartesian product simplifies the fixpoint equation to Equation (3).

$$\mathbf{Stream}(\mathbf{X}; \mathbf{Y}) = \mathbf{hom}(X_0, Y_0) \times \mathbf{Stream}(X_0 \times X_1, X_2, \dots; Y_1, Y_2, \dots). \tag{3}$$

**Theorem 3.1.** *The greatest fixpoint of Equation (3) is given by*

$$\mathbf{Stream}(\mathbf{X}; \mathbf{Y}) = \prod_{n \in \mathbb{N}} \mathbf{hom}(X_0 \times \dots \times X_n, Y_n). \tag{4}$$

*Remark* 3.2. The category $\mathbf{Stream}_{\mathbb{C}}$, when $\mathbb{C}$ is cartesian monoidal, coincides with the coKleisli category for a comonad $\mathbf{List}^+ : \mathbb{C}^{\mathbb{N}} \to \mathbb{C}^{\mathbb{N}}$ defined by $\mathbf{List}^+(\mathbf{X})_n \coloneqq \prod_{i=0}^{n} X_i$, which is analogous to the coalgebraic *causal stream functions* of [27]. A stream with types $X_0, X_1, \dots$ can be recovered as a stream transducer with no inputs. That is, an element of the set $\mathbf{Stream}(1, 1, \dots; X_0, X_1, \dots)$.

**Stochastic processes.** Let $\mathbf{Stoch}$ be the category of stochastic functions, i.e. the Kleisli category of the finite distribution monad $\mathbf{D} : \mathbb{Set} \to \mathbb{Set}$. A stochastic process is usually defined as a sequence of random variables indexed by time [23]; that is, a sequence of distributions $p_n \in \mathbf{D}(Y_0 \times \dots \times Y_n)$ that are compatible under marginalisation, $p_{n+1} ; \pi_{Y_0, \dots, Y_n} = p_n$. We give a slightly more general notion that allows for the stochastic process to be controlled by an input.

**Definition 3.3** (Stochastic process)**.** Let $\mathbf{X} = X_0, X_1, \dots$ and $\mathbf{Y} = Y_0, Y_1, \dots$ be families of sets. A *stochastic process* from $\mathbf{X}$ to $\mathbf{Y}$ is a sequence of functions, $f_n : X_0 \times \dots \times X_n \to \mathbf{D}(Y_0 \times \dots \times Y_n)$ for each $n \in \mathbb{N}$, such that $f_n$ coincides with the marginal distribution of $f_{n+1}$ on the first $n$ variables. In other words, $f_{n+1} ; \pi_{Y_0, \dots, Y_n} = \pi_{X_0, \dots, X_n} ; f_n$.

**Proposition 3.4.** *Stochastic processes form a category* $\mathbf{StochProc}$ *with composition and identities defined component-wise by composition and identities in* $\mathbf{Stoch}$*.*

Stochastic processes are precisely stream transducers in the category of stochastic functions.

**Theorem 3.5.** *There exists an isomorphism of categories* $\mathbf{StochProc} \cong \mathbf{Stream}_{\mathbf{Stoch}}$*.*

# 4 Conclusion and future work

We defined a category whose morphisms are stream transducers and we characterized it in terms of final coalgebras. We recovered the usual notion of stream transducers in the cartesian case [25, 27]. In the case of the category of stochastic functions, we recovered a notion of controlled stochastic process [23]. This category could be used as semantics for probabilistic stream-based programming languages [12, 28]. We would like to investigate what this construction corresponds to in the case of other monoidal categories, like those of partial maps [7], non-deterministic maps [4] and quantum processes [1, 5].

# References


[1] Samson Abramsky and Bob Coecke. Categorical quantum mechanics. *Handbook of quantum logic and quantum structures*, 2:261–325, 2009. arXiv:0808.1023.

[2] Jiří Adámek. Free algebras and automata realizations in the language of categories. *Commentationes Mathematicae Universitatis Carolinae*, 015(4):589–602, 1974. URL: http://eudml.org/doc/16649.

[3] Filippo Bonchi, Jens Seeber, and Pawel Sobocinski. Graphical conjunctive queries. In Dan R. Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*, volume 119 of *LIPIcs*, pages 13:1–13:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CSL.2018.13.

[4] Manfred Broy and Gheorghe Ştefănescu. The algebra of stream processing functions. *Theoretical Computer Science*, 258(1-2):99–129, 2001.

[5] Titouan Carette, Marc de Visme, and Simon Perdrix. Graphical language with delayed trace: Picturing quantum computing with finite memory. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470553.

[6] Kenta Cho and Bart Jacobs. Disintegration and Bayesian Inversion via String Diagrams. *Mathematical Structures in Computer Science*, pages 1–34, March 2019. arXiv:1709.00322, doi:10.1017/S0960129518000488.

[7] J Robin B Cockett and Stephen Lack. Restriction categories i: categories of partial maps. *Theoretical computer science*, 270(1-2):223–259, 2002.

[8] Elena Di Lavore, Alessandro Gianola, Mario Román, Nicoletta Sabadini, and Paweł Sobociński. A canonical algebra of open transition systems. *arXiv preprint*, 2020. arXiv:2010.10069.

[9] Dale Dougherty and Arnold Robbins. *sed & awk: Unix power tools*. O'Reilly, 1997.

[10] Tobias Fritz. A synthetic approach to markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, 2020. URL: http://arxiv.org/abs/1908.07021, arXiv:1908.07021.

[11] Richard Garner. Stream processors and comodels. *arXiv preprint arXiv:2106.05473*, 2021.

[12] Nicolas Halbwachs, Fabienne Lagnier, and Christophe Ratel. Programming and verifying real-time systems by means of the synchronous data-flow language LUSTRE. *IEEE Trans. Software Eng.*, 18(9):785–793, 1992. doi:10.1109/32.159839.

[13] Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*, volume 59 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2016. doi:10.1017/CBO9781316823187.

[14] André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in mathematics*, 88(1):55–112, 1991.

[15] Piergiulio Katis, Nicoletta Sabadini, and Robert F. C. Walters. Feedback, trace and fixed-point semantics. *RAIRO Theor. Informatics Appl.*, 36(2):181–194, 2002. doi:10.1051/ita:2002009.

[16] J. Lambek. Cartesian closed categories and typed $\lambda$-calculi. In Guy Cousineau, Pierre-Louis Curien, and Bernard Robinet, editors, *Combinators and Functional Programming Languages*, Lecture Notes in Computer Science, pages 136–175, Berlin, Heidelberg, 1986. Springer. doi:10.1007/3-540-17184-3_44.

[17] Joachim Lambek. A fixpoint theorem for complete categories. *Mathematische Zeitschrift*, 103(2):151–161, 1968.

[18] Fosco Loregian. *(Co)end Calculus*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2021. doi:10.1017/9781108778657.

[19] Saunders Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer Verlag, 1971. doi:10.1007/978-1-4757-4721-8.

[20] Evan Patterson, Ioana Baldini, Aleksandra Mojsilovic, and Kush R. Varshney. Teaching machines to understand data science code by semantic enrichment of dataflow graphs. *CoRR*, abs/1807.05691, 2018. arXiv:1807.05691.

[21] Dirk Pattinson, Peter Hancock, and Neil Ghani. Representations of stream processors using nested fixed points. *Logical Methods in Computer Science*, 5, 2009. URL: arXiv:0905.4813.

[22] Mario Román. Comb diagrams for discrete-time feedback. *CoRR*, abs/2003.06214, 2020. arXiv:2003.06214.

[23] Sheldon M. Ross. *Stochastic processes*, volume 2. John Wiley & Sons, 1996.

[24] Peter Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010.

[25] David Sprunger and Shin-ya Katsumata. Differentiable causal computations via delayed trace. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–12. IEEE, 2019. doi:10.1109/LICS.2019.8785670.

[26] Gheorghe Ştefănescu. *Feedback Theories (a Calculus for Isomorphism Classes of Flowchart Schemes)*. Inst. de Mat., 1986.

[27] Tarmo Uustalu and Varmo Vene. Comonadic notions of computation. In Jirí Adámek and Clemens Kupke, editors, *Proceedings of the Ninth Workshop on Coalgebraic Methods in Computer Science, CMCS 2008, Budapest, Hungary, April 4-6, 2008*, volume 203 of *Electronic Notes in Theoretical Computer Science*, pages 263–284. Elsevier, 2008. doi:10.1016/j.entcs.2008.05.029.

[28] William W Wadge, Edward A Ashcroft, et al. *Lucid, the dataflow programming language*, volume 303. Academic Press London, 1985.

# A Appendix

## A.1 Omitted proofs

**Theorem A.1** (Previous theorem 3.1)**.** *Let $\mathbb{C}$ be a cartesian monoidal category. The set of stream transducers over $\mathbb{C}$ is characterized by*

$$\mathbf{Stream}(X_0, X_1, \dots; Y_0, Y_1, \dots) = \prod_{n \in \mathbb{N}} \mathbf{hom}(X_0 \times \dots \times X_n, Y_n). \tag{5}$$

*Proof sketch.* We apply the universal property of the cartesian product to simplify the hom-profunctor. We then apply the technique of Yoneda reduction [18] to simplify the coend expression.

$$\begin{aligned}
&\mathbf{Stream}(X_0, X_1, \dots; Y_0, Y_1, \dots) \\
&\quad = \quad \{\text{by Theorem 2.3}\} \\
&\int^M \mathbf{hom}(X_0, Y_0 \times M) \times \mathbf{Stream}(M \times X_1, X_2, \dots; Y_1, Y_2, \dots) \\
&\quad \cong \quad \{\text{by the universal property of the product}\} \\
&\int^M \mathbf{hom}(X_0, Y_0) \times \mathbf{hom}(X_0, M) \times \mathbf{Stream}(M \otimes X_1, X_2, \dots; Y_1, Y_2, \dots) \\
&\quad \cong \quad \{\text{simplify the coend using a Yoneda reduction [18]}\} \\
&\mathbf{hom}(X_0, Y_0) \times \mathbf{Stream}(X_0 \times X_1, X_2, \dots; Y_1, Y_2, \dots)
\end{aligned} \tag{6}$$

By Lambek's theorem [17], the final coalgebra of a functor is also its greatest fixpoint. By Adamek's theorem [2], the final coalgebra of a functor can be computed as the projective limit of applying the functor repeatedly to a terminal object. □

**Theorem A.2** (Previous theorem 3.5)**.** *There exists an isomorphism of categories* $\mathbf{StochProc} \cong \mathbf{Stream}_{\mathbf{Stoch}}$*. Stochastic processes are precisely stream transducers on the category of stochastic functions.*

*Proof sketch.* The memory channels will contain all the previous inputs and outputs: $M_t = X_0 \times \dots \times X_t \times Y_0 \times \dots Y_t$. We use that every family of functions describing a stochastic process gives rise to a family of distributions $c_n \colon X_0 \times \dots \times X_n \times Y_0 \times \dots \times Y_{n-1} \to Y_n$, called conditional distributions in the context of Markov categories [10]. In order to show that this actually defines a bijection, we need to use some properties of this factorization and properties of the coend that describes streams that we do not detail here. □

## A.2 Coend Calculus

*Coend calculus* is the name given to the a branch of category theory that describes the behaviour of certain colimits called *coends*. We follow the standard presentation of coend calculus from [18].

**Definition A.3.** *Coends* are defined as the coequalizers of the action of morphisms on both arguments of a profunctor.

$$\mathrm{coend}(P) \coloneqq \mathrm{coeq}\left( \textstyle\coprod_{f \colon B \to A} P(A,B) \Longrightarrow \coprod_{X \in \mathbb{C}} P(X,X) \right).$$

Coends are usually denoted with a superscripted integral, drawing on an analogy with the classical calculus.

$$\int^{X \in \mathbb{C}} P(X,X) \coloneqq \mathrm{coend}(P)$$

**Proposition A.4** (*CoYoneda* reduction)**.** *Let* $\mathbb{C}$ *be any category and let* $F \colon \mathbb{C} \to \mathbb{S}\mathrm{et}$ *be what is usually called a co-presheaf; the following isomorphism holds for any given object* $A \in \mathbb{C}$*.*

$$\int^{X \in \mathbb{C}} \mathbf{hom}(X,A) \times FX \cong FA.$$

*Following the analogy with classical analysis, the* **hom** *works as a Dirac's delta.*

**Proposition A.5** (*Fubini rule*)**.** *Coends commute between them; that is, there exists a natural isomorphism*

$$\int^{X_1 \in \mathbb{C}} \int^{X_2 \in \mathbb{C}} P(X_1,X_2,X_1,X_2) \cong \int^{X_2 \in \mathbb{C}} \int^{X_1 \in \mathbb{C}} P(X_1,X_2,X_1,X_2).$$

*In fact, they are both isomorphic to the coend over the product category,*

$$\int^{(X_1,X_2) \in \mathbb{C} \times \mathbb{C}} P(X_1,X_2,X_1,X_2).$$

*Following the analogy with classical analysis, coends follow the Fubini rule for integrals.*

## A.3 Profunctors

**Definition A.6.** A *profunctor* from a category $\mathbb{A}$ to a category $\mathbb{B}$ is a functor $P\colon \mathbb{A}^{op} \times \mathbb{B} \to \mathbb{Set}$.

**Definition A.7** (Sequential composition)**.** Two profunctors $P\colon \mathbb{A}^{op} \times \mathbb{B} \to \mathbb{Set}$ and $Q\colon \mathbb{B}^{op} \times \mathbb{C} \to \mathbb{Set}$ compose *sequentially* into a profunctor $P \diamond Q\colon \mathbb{A}^{op} \times \mathbb{C} \to \mathbb{Set}$ defined by

$$(P \diamond Q)(A, C) \coloneqq \int^{B \in \mathbb{B}} P(A, B) \times Q(B, C).$$

The *hom-profunctor* **hom**: $\mathbb{A}^{op} \times \mathbb{A} \to \mathbb{Set}$ that returns the set of morphisms between two objects is the unit for sequential composition. Sequential composition is associative up to isomorphism.

**Definition A.8** (Parallel composition)**.** Two profunctors $P\colon \mathbb{A}_1^{op} \times \mathbb{B}_1 \to \mathbb{Set}$ and $Q\colon \mathbb{A}_2^{op} \times \mathbb{B}_2 \to \mathbb{Set}$ compose *in parallel* into a profunctor $P \times Q\colon \mathbb{A}_1^{op} \times \mathbb{A}_2^{op} \times \mathbb{B}_1 \times \mathbb{B} \to \mathbb{Set}$ defined by

$$(P \times Q)(A, A', B, B') \coloneqq P(A, B) \times Q(A', B').$$

**Definition A.9** (Communicating profunctor composition)**.** Let $\mathbb{A}, \mathbb{B}, \mathbb{C}$ be categories and let $\mathbb{B}$ have a monoidal structure. Two profunctors $P\colon \mathbb{A}^{op} \times \mathbb{B} \to \mathbb{Set}$ and $Q\colon \mathbb{B}^{op} \times \mathbb{C} \to \mathbb{Set}$ *compose communicating along* $\mathbb{B}$ into the profunctor $(P \odot Q)\colon \mathbb{A}^{op} \times \mathbb{B} \times \mathbb{B}^{op} \times \mathbb{C} \to \mathbb{Set}$ defined by

$$(P \odot Q)(A, B; B', C) \coloneqq \int^{M} P(A, B \otimes M) \times Q(M \otimes B', C).$$

The profunctors $\mathbf{hom}(I, \bullet)\colon \mathbb{B} \to \mathbb{Set}$ and $\mathbf{hom}(\bullet, I)\colon \mathbb{B}^{op} \to \mathbb{Set}$ are left and right units with respect to communicating composition. The communicating composition of three profunctors $P\colon \mathbb{A}^{op} \times \mathbb{B} \to \mathbb{Set}$, $Q\colon \mathbb{B}^{op} \times \mathbb{C} \to \mathbb{Set}$ and $R\colon \mathbb{C}^{op} \times \mathbb{D} \to \mathbb{Set}$ is associative up to isomorphism and a representative can be written simply by $(P \odot Q \odot R)\colon \mathbb{A}^{op} \times \mathbb{B} \times \mathbb{B}^{op} \times \mathbb{C} \times \mathbb{C}^{op} \times \mathbb{D} \to \mathbb{Set}$, where both $\mathbb{B}$ and $\mathbb{C}$ are assumed to have a monoidal structure.

## A.4 Stochastic functions

**Definition A.10.** The finite distribution commutative monad $\mathbf{D}\colon \mathbb{Set} \to \mathbb{Set}$ is associates to each set the set of finite-support probability distributions over that set.

$$\mathbf{D}(X) \coloneqq \left\{ p\colon X \to [0,1] \;\middle|\; \#\{x \mid p(x) > 0\} < \infty;\ \text{and} \sum_{p(x)>0} p(x) = 1 \right\}.$$

We call **Stoch** to the symmetric monoidal kleisli category of the finite distribution monad.